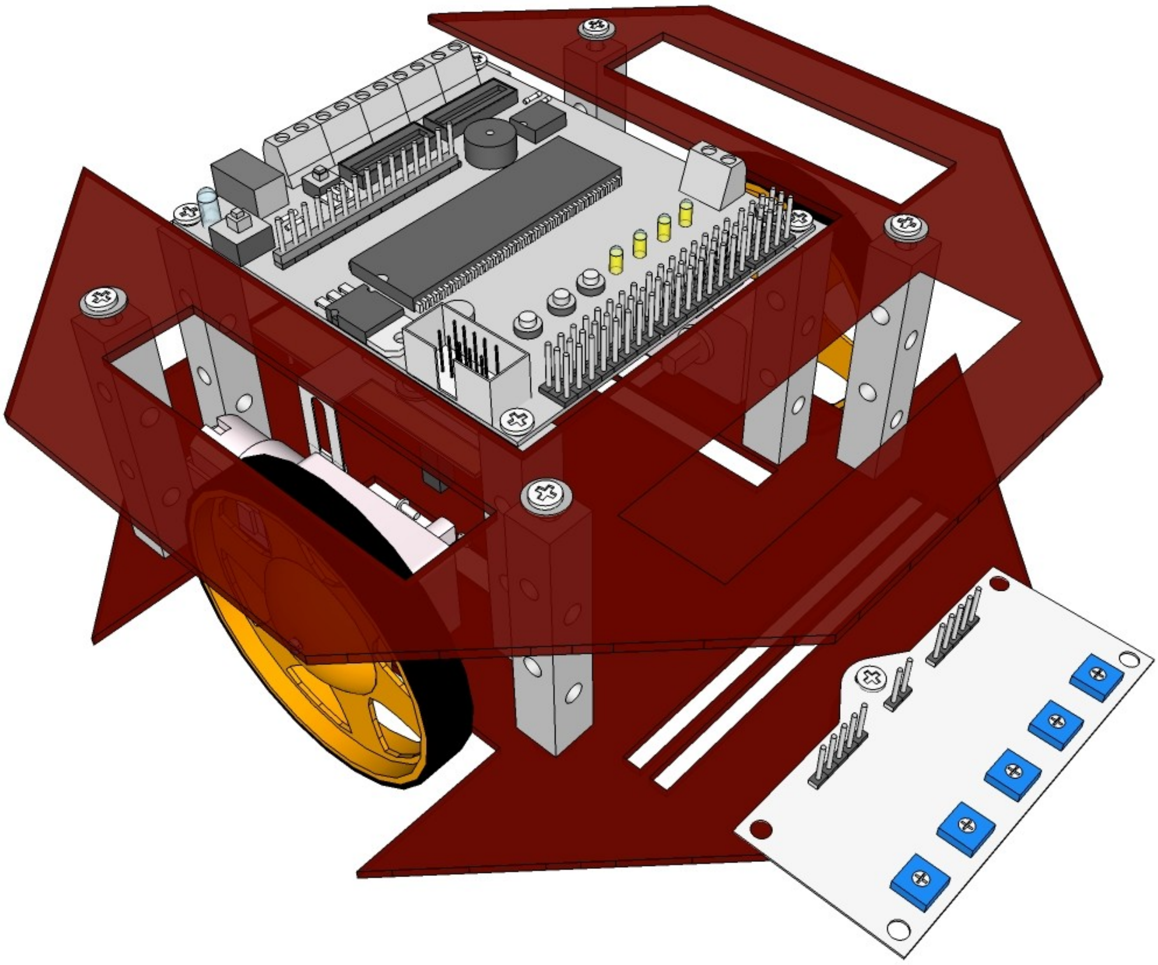
The Zed Guide

An AVR based autonomous robotic platform



Welcome to the world of autonomous robotics!

About Zed

We realize that the current trend of autonomous robotics competitions and hobby robotics are tending towards line follower based robots. Zed is a platform specialized to work in this field. Although a general purpose platform, the structure of the Zed has been optimized keeping in mind, Line following based applications

With this kit the user shall have a hands-on experience of working with Line sensor arrays, different actuators like DC geared and stepper motors, their drivers, LCD interfacing and serial communication with the PC.

Zed follows a modular approach throughout its construction, so the user can easily add new features and behaviors from line following, Grid solving, Line Maze solving and a basic micromouse. It can also be used as a generic embedded systems development platform.

We hope that building this autonomous robot will be an enriching experience and would drive you to further pursue robotics seriously.

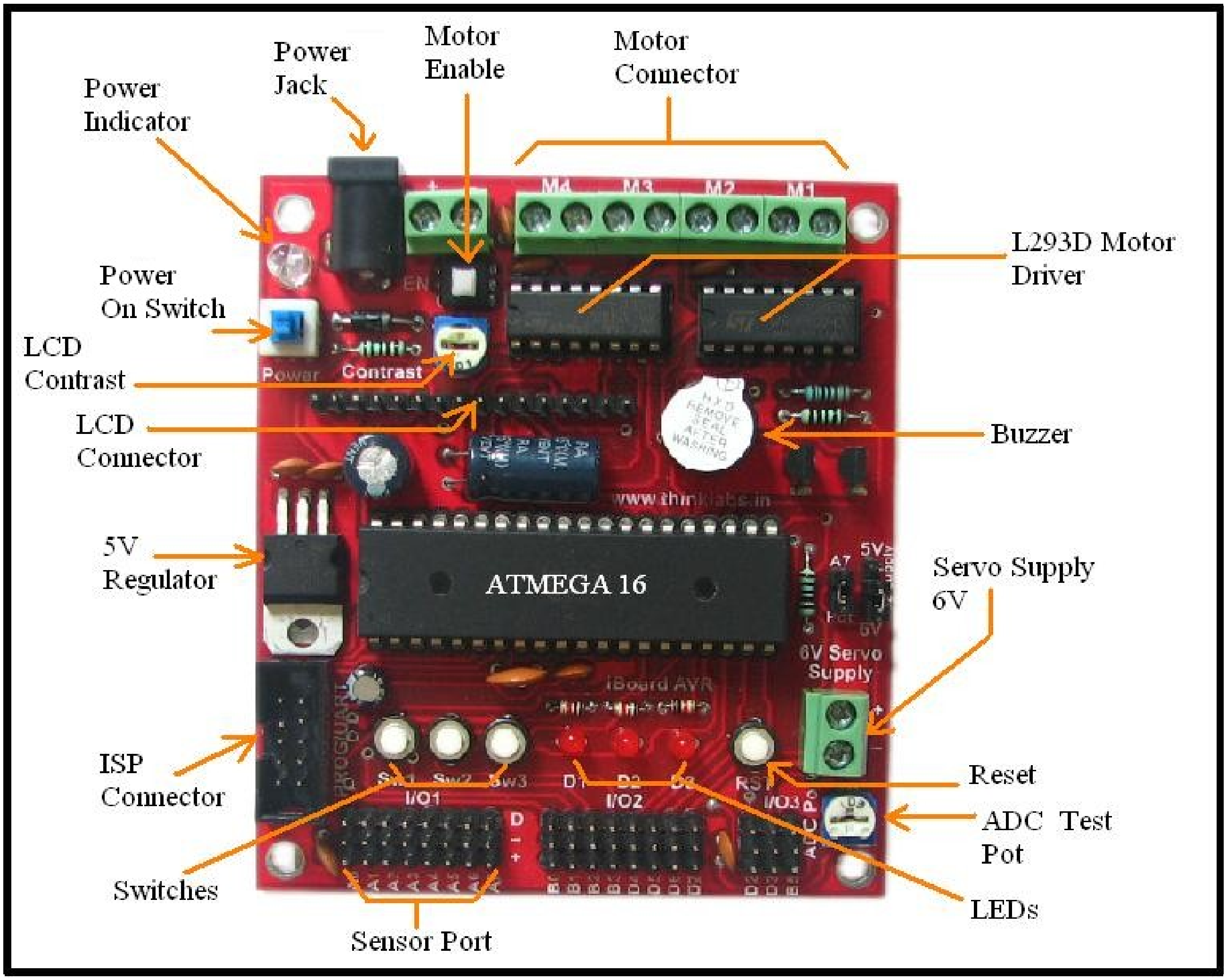
The Zed Controller Board

The Zed Controller Board is based around the powerful AVR microcontroller.

Features:

- Includes ATMEL AT Mega 16 (Also supports ATMega32) Microcontroller with 16kB flash memory working at 16Mips
- In system programmable. On board programmer
- On board regulated power supply. Power indicator LED
- 2 on board Dual full H bridge motor driver for 2 stepper or 4 DC motors
- Separate ON/OFF switch for power and motor drivers

- 3 LEDs for status or debugging purposes
- 3 ON/OFF switches for external inputs/interrupts
- On board LCD connector(multipurpose port, which can be used for other applications also)



Parts identification:

- Power On Switch: It's a basic push to on - push to off type switch.
- IC 7805: It's a three terminal linear 5 volt regulator used to supply the microcontroller and other peripherals.
- Motor Enable switch: This switch is used to enable/disable the motor driver chips hence in turn enabling/disabling the motors.
- Reset Switch: This switch is used to reset the microcontroller.
- L293D: It is a 4 channel motor driver with 600mA of current per channel and has inbuilt clamp diodes. The board contains two such ICs.
- Potentiometer (Pot): The potentiometer is used to vary the contrast of the LCD.
- Sensor port: At a time, 8 individual sensor modules can be connected to this

port. The port also provides a 5V supply needed drive the sensors.

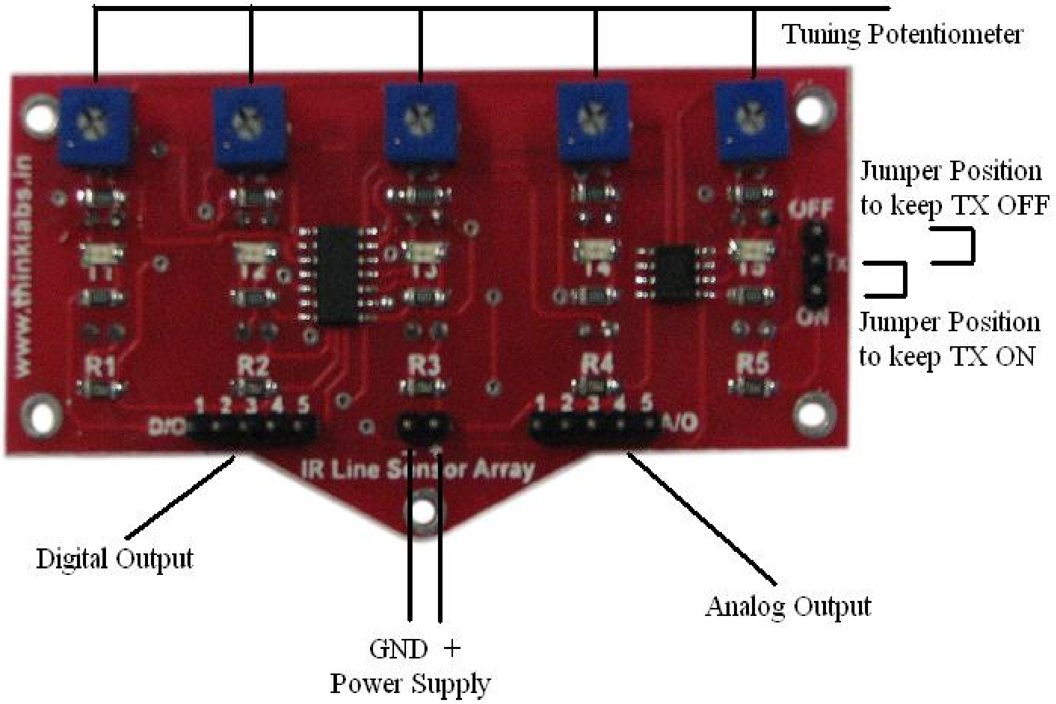
- ISP connector: This is a 10 pin connector used to connect to the PC's COM port through AVR ISP Programmer for programming or for general UART communications.
- Switch array: 3 general purpose switches are connected in the active-low configuration.
- LEDs: 3 LEDs connected to the pins of microcontroller.
- Crystal: A crystal sets the microcontroller's clock frequency to 16 MHz.
- Beeper: Connected in the active low mode, the beeper can easily be used to get audible feedbacks from the controller.
- ADC Test: Short the two pins of the jumper next to the potentiometer and the potentiometer gets connected to the PortA 7.
- LCD Connector: To attach 16x2 LCD.

Board Connection Details:

Port A	0 to 7	Sensor Connector
Port B	0 to 3	Motor Driver Inputs (M3 & M4)
	4	LED D3
	5	Switch 3
	6	LED D2
	7	Programmer
Port C	0	LCD Control PIN (RS)
	1	LCD Backlight Control PIN
	2	LCD Control PIN (E)
	3	Buzzer
	4 to 7	LCD Data PINS
Port D	0	USART RX
	1	USART TX + LED D1
	2	Switch 1
	3	Switch 2
	4 to 7	Motor Driver Input (M1 & M2)

Note: Switches are Active Low, LEDS are Active High

The Line Sensor Array Connection Diagram



Sample Programs

Blinking an LED

```
#include<avr/io.h>
#include<util/delay.h>

int main(void)
{
DDRD = 0xFF;
while(1)
	{
	PORTD = PORTD^0x02;
	_delay_ms(1000);
	}
}
```

Controlling LEDs with Switches

```
#include<avr/io.h>
#include<util/delay.h>
```

```
#define SW1 !(PIND & 0x04)  //Check if      SW1 is pressed
#define SW2 !(PIND & 0x08)  //       "                 SW2    "
#define SW3 !(PINB & 0x10)  //       "                 SW3    "
#define LED1 0x02                    //Position of LED1 on PORTD
#define LED2 0x40                    //       "                 LED2 on PORTB
#define LED3 0x10                    //   "              LED3 on PORTB

int main(void)
{
DDRB = 0xDF;  // Make 4th pin an input port
PORTB = 0x20;// Enable Pull Up on the 5th pin
DDRD = 0xF3;  // Make 2nd and 3rd pins an input port
PORTD = 0x0C;          // Enable Pull Up on the 2nd and 3rd Pins

while(1)
        {
        if(SW1)          // If SW=1, Switch is not pressed. If SW=0, Switch is pressed
                {
                PORTD = PORTD | LED1;       //Turn LED on without disturbing other pins
                }
        else
                {
                PORTD = PORTD & ~LED1;     //Turn LED off without disturbing other pins
                }
        if(SW2)
                {
                PORTB = PORTB | LED2;
                }
        else
                {
                PORTB = PORTB & ~LED2;
                }
        if(SW3)
                {
                PORTB = PORTB | LED3;
                }
        else
                {
                PORTB = PORTB & ~LED3;
                }
        }
}
```

Running a Unipolar stepper Motor

```
/* Connect a Unipolar stepper motor to M3-M4 PTR Connectors. Common terminal goes to ground.
In this Example, the Unipolar motors run in single phase mode./*

#include<avr/io.h>
#include<util/delay.h>

int main(void)

DDRB = 0x0f;            //Make Motor pins of PORT B outputs

while(1)
        {
        PORTB = 0x08;// Take one step
        _delay_ms(20);          //Delay
        PORTB = 0x04;//take the next step
        _delay_ms(20);          //Delay
        PORTB = 0x02;//take the next step
        _delay_ms(20);          //Delay
        PORTB = 0x01;//take the next step
        _delay_ms(20);          //Delay
        }
```

Grid Solver

Write a program to reach the diagonally opposite corner of the grid

```
/*Connect the Sensors to PortA Pin 0 (Rightmost Sensor) to Pin 4 (Leftmost Sensor)*/

#include<avr/io.h>
#include<util/delay.h>
#include"stepper.h"

/* Following are macros which are used for condition checking. Each macro check for a specific
condition. Read the operations properly to understand how they work. */

#define STLINE ((PINA&0x11)==0x11)&((PINA&0x1f)!=0x1f)
#define IDEAL ((PINA&0x1F)==0x1B)
#define LINELEFT ((PINA&0x1f)==0x03)|((PINA&0x1f)==0x07)
#define LINERIGHT ((PINA&0x1f)==0x18)|((PINA&0x1f)==0x1c)
#define LEFT90 ((PINA&0x1f)==0x07)|((PINA&0x1f)==0x03)|((PINA&0x1f)==0x01)
#define RIGHT90 ((PINA&0x1f)==0x1c)|((PINA&0x1f)==0x18)|((PINA&0x1f)==0x10)
#define JUNC !(PINA&0x1f)

#define SPEED 30
```

```
int main(void)
{
unsigned char turn=0, s;        //turn flag remembers the last turn taken. S is to count number of
                                steps
DDRB = 0x0f;    DDRD = 0xf0;    DDRA = 0x00;  PORTA = 0xff;  //Initialize motor and Sensor Port

while(1)
        {
        while(STLINE)   //Follow Straight Line
                {
                if(IDEAL)       //Ideal Condition
                        {
                        lstep_f(SPEED);                 //Go Straight
                        rstep_f(SPEED);
                        }
                if(LINELEFT)    //Turn Left
                        {
                        lstep_f(SPEED);
                        }
                if(LINERIGHT)   //Turn Right
                        {
                        rstep_f(SPEED);
                        }
                }
        if(LEFT90)      //90 degree LEFT TURN
                {
                for(s=0;s<=3;s++)       //Go 4 steps and recheck condition
                        {
                        lstep_f(SPEED);
                        rstep_f(SPEED);
                        }
                if(LEFT90)                      // If condition is still LEFT90, do this
                        {
                        for(s=0;s<=20;s++)      //Go Forward 21 Steps
                                {
                                lstep_f(SPEED);
                                rstep_f(SPEED);
                                }
                        for(s=0;s<=10;s++)      //Diff Turn Left 11 Steps
                                {
                                lstep_f(SPEED);
                                rstep_r(SPEED);
                                }
                        while(!(IDEAL|LINELEFT|LINERIGHT))              //Diff turn Left until line is met
                                {
                                lstep_f(SPEED);
                                rstep_r(SPEED);
                                }
                        turn=0;                         //Set Turn Flag = Left (0)
                        }
                }
        if(RIGHT90)     //90 degree RIGHT TURN
        {
        for(s=0;s<=3;s++)       //Go 4 steps and recheck condition
            {
            lstep_f(SPEED);
            rstep_f(SPEED);
            }
        if(RIGHT90)                     // If condition is still Right90, do this
            {
            for(s=0;s<=20;s++)  //Go Forward 21 Steps
                {
                lstep_f(SPEED);
                rstep_f(SPEED);
                }
            for(s=0;s<=10;s++)  //Diff Turn Left 11 Steps
                {
                rstep_f(SPEED);
                lstep_r(SPEED);
                }
            while(!(IDEAL|LINELEFT|LINERIGHT))          //Diff turn Left until line is met
                {
                rstep_f(SPEED);
                lstep_r(SPEED);
                }
            turn=1;                         //Set Turn Flag = Right (1)
            }
        }
    if (JUNC)           //If Junction is detected,
        {
        for(s=0;s<=20;s++)      //Go straight 21 Steps
            {
            lstep_f(SPEED);
            rstep_f(SPEED);
            }
        if (turn==1)    //If last turn was right,
            {
            for(s=0;s<=10;s++)  //11 Steps Diff Left
                {
                lstep_f(SPEED);
                rstep_r(SPEED);
                }
            while(!(IDEAL|LINELEFT|LINERIGHT))          //Diff turn Left until line is met
                {
                lstep_f(SPEED);
                rstep_r(SPEED);
                }
            turn=0;                     //Set Turn Flag = Left (0)
            }
        else                                //If last turn was Left
            {
            for(s=0;s<=20;s++)  //11 Steps Diff Right
                {
                lstep_r(SPEED);
                rstep_f(SPEED);
                }
            while(!(IDEAL|LINELEFT|LINERIGHT))          //Diff turn Right until line is met
                {
                rstep_f(SPEED);
                lstep_r(SPEED);
                }
            turn=1;                     //Set Turn Flag = Right (1)
            }
        }
    }

}
```

Writing a string to EEPROM

```
#include<avr/io.h>
#include<util/delay.h>

unsigned char add = 0,d = 0;
unsigned char data[10] = {2,0,2,0,2,0,2,0,2,0};         //The String to be Written


int main(void)
{
for (add=0;add<10;add++)        //Keep doing while the string ends
    {
    EEAR = add;                 //Set address to be written to
    EEDR = data[d];             //Set data to be written
    EECR = 0x04;                //Set the EEMWE Bit
    EECR = 0x02;                //Start writing by setting the EEWE Bit
    while(EECR & 0x02);         //Wait while EEWE Bit is reset (Write finish)
    d++;                        //Go to the next string character
    }

while(1)
    {}                          //Do nothing
}
```

Reading from EEPROM

```
#include<avr/io.h>
#include<util/delay.h>

unsigned char add = 0,dat;

int main(void)
{
DDRD = 0x02;            //Initialize an LED PIN as output
PORTD = 0x00;
while(add<10)
        {
        EEAR = add;             //Set Address to be read from
        EECR = 0x01;            //Set the EERE bit to begin writing
        dat = EEDR;             //Store the data in a variable
        if (dat)                // If data is non zero, glow the LED
                {
                PORTD = 0x02;
                }
        else
                {
                PORTD = 0x00;
                }
        _delay_ms(2000);        //Wait for 2 seconds
        add++;                  //Increment the address value
        }
while(1);                       //Do nothing


}  // End of code ☺
```

For any queries Contact: techqueries@thinklabs.in

Or visit forums at www.thinklabs.in/forums

Read related articles at www.thinklabs.in/resources

For buying components, visit www.thinklabs.in/roboshop